Remarks

Applicant respectfully requests reconsideration of this application as amended. No claims have been amended, cancelled, or added. Therefore, claims 2-6, 8-12, 14-18, and 21-25 are presented for examination.

35 U.S.C. §103(a) Rejection

Claims 2-6, 8-12, 14-18 and 21-25 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Cramer et al. (U.S. Patent No. 5,107,418) in view of Archambault (U.S. Patent No. 6,173,444). Applicant submits that the present claims are patentable over Cramer in view of Archambault.

Cramer discloses a method for representing scalar data dependencies for an optimizing compiler wherein a global scalar data dependence graph is created to represent all of the scalar objects in an entire program. More specifically, scalar data dependencies are represented by a use-definition chain, a definition-use chain, or a definition-definition chain. Also, the representation of scalar data dependencies is created for the entire program and is maintained during the entire compilation or assembly of the program. (Cramer at col. 2, ll. 5-25.)

Archambault discloses a method that reduces the size of alias sets associated with program pointers through the use of a pointer alias graph. Standard data flow gathering techniques are used to develop the pointer alias graph. The nodes in the graph represent either a definition of a pointer variable or a use of a pointer variable, and each node has an associated alias set. The initial alias sets for definition nodes is the right hand side of the pointer variable assignment operation, and the initial alias set for use nodes is the value of the

object at that execution point. Location information, the basic block number (relative to the flow graph) and position within the basic block, is saved for each node. (Archambault at col. 5, ll. 4-17).

Claim 2 recites:

> A computer-implemented method, comprising:
> assigning a definition-node for one or more definition statements in an intermediate language program;
> assigning a use-node for one or more use statements in the intermediate language program;
> assigning an alias-node for one or more aliases representing an equivalence class of memory accesses;
> introducing an edge into a dependence flow graph connecting each definition-node to the alias-node corresponding to the alias representing the equivalence class to which the definition-node belongs; and
> introducing an edge in the dependence flow graph connecting each use-node to the alias-node corresponding to the alias representing the equivalence class to which the use-node belongs,
> wherein a number of the edges in the dependence flow graph is linear to a number of the nodes in the dependence flow graph, and wherein the number of edges is independent of a definition-use structure of the intermediate language program.

Applicant submits that Cramer does not disclose or suggest a number of edges in a dependence flow graph being independent of a definition-use structure of an intermediate language program, as recited by claim 2. The Examiner states that Cramer does not explicitly disclose such a feature. (Office Action mailed 11/2/05 at pgs. 3 & 4.) Instead, the Examiner provides arguments as to how Cramer implicitly discloses this feature. For instance, the Examiner states:

> As each of the definition or use node is created for a variable in the flow graph, Cramer teaches creating an edge connecting a alias corresponding to a equivalence class associating a definition node (step 615, 617, 618: YES – Fig. 6) and creating an edge in a dependence flow graph connecting a alias corresponding to a equivalence class associating a use node (steps 616: YES, step 615 – Fig. 6), hence creating edges only for those aliases corresponding to a equivalence class (Note: the number of aliases thus created is not directly proportional to the number of def-use structures, i.e.

the number of edge introducing as in Note* is independent from the number of definition/use structures being defined for variables in the flow graph from above – as in Fig. 6).

Id. at pg. 4. The Examiner relies on three steps in a Figure to disclose the cited feature of claim 2. The Figure 6 of Cramer shows a method employed for processing pointer deferences, and how these are added to a data dependence graph. However, the process of Figure 6 of Cramer only discloses examining dependence nodes in a basic block for aliases between each other. There is no discussion in Figure 6 of a separate alias node that the dependence nodes connect to. For example, the process of Figure 6 extracts alias information from a dependence node, also called the "current node" in the context of the Figure 6 discussion (steps 613 and 614). Then, the process loops through the other dependence nodes in the basic block to determine whether they alias to the current node (step 615). If there is an alias between another dependence node and the current node that is legal, ***that aliased dependence node is connected to the current node*** (step 617). (Cramer at col. 8, ll. 23-41.)

Accordingly, Figure 6 of Cramer does not disclose the process and structure of claim 1 where definition nodes of a particular equivalence class are connected to an alias node for that equivalence class. Rather, dependence nodes in Cramer are connected to each other and not to a separate alias node. Figure 6, in fact, discloses the opposite of the cited feature of claim 1 – it discloses that the number of edges in its dependence flow graph is dependent on the definition-use structure of the program. Accordingly, Cramer does not disclose or suggest a number of edges in a dependence flow graph being independent of a definition-use structure of an intermediate language program.

Nor does Archambault disclose or suggest a number of edges in a dependence flow graph being independent of a definition-use structure of an intermediate language program. As neither Cramer nor Archambault individually disclose or suggest the cited feature of claim 2, any combination of Cramer and Archambault also does not disclose or suggest such a feature. Therefore, claim 2 is patentable over Cramer in view of Archambault. Claims 3-6 and 23 depend from claim 2 and include additional limitations. Therefore, claims 3-6 and 23 are also patentable over Cramer in view of Archambault.

Claims 8, 14, and 21, as amended, each recite, in part, a number of edges in a dependence flow graph being independent of a definition-use structure of an intermediate language program. As discussed above, neither Cramer nor Archambault disclose or suggest such a feature. Therefore, claims 8, 14, and 21, as well as their respective dependent claims, are patentable over Cramer in view of Archambault for the reasons discussed above with respect to claim 1.

Applicant respectfully submits that the rejections have been overcome and that the claims are in condition for allowance. Accordingly, applicant respectfully requests the rejections be withdrawn and the claims be allowed.

The Examiner is requested to call the undersigned at (303) 740-1980 if there remains any issue with allowance of the case.

Applicant respectfully petitions for an extension of time to respond to the outstanding Office Action pursuant to 37 C.F.R. § 1.136(a) should one be necessary. Please charge our Deposit Account No. 02-2666 to cover the necessary fee under 37 C.F.R. § 1.17(a) for such an extension.

Please charge any shortage to our Deposit Account No. 02-2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: January 23, 2006

Ashley R. Ott
Reg. No. 55,515

12400 Wilshire Boulevard
7th Floor
Los Angeles, California 90025-1026
(303) 740-1980